The Journal of Supercomputing (2019) 75:7573–7591
https://doi.org/10.1007/s11227-019-02971-x

An investigation of latency prediction for NoC-based communication architectures using machine learning techniques

Jefferson Silva[1] · Márcio Kreutz[2] · Monica Pereira[2] · Marjory Da Costa-Abreu[2]

Published online: 19 August 2019

Abstract

Due to the increasing number of cores in Systems on Chip (SoCs), bus architectures have suffered with limitations regarding performance. As applications demand higher bandwidth and lower latencies, buses have not been able to comply with such requirements due to longer wires and increased capacitance. Facing this scenario, Networks on Chip (NoCs) emerged as a way to overcome the limitations found in bus-based systems. Fully exploring all possible NoC characteristics settings is unfeasible due to the vast design space to cover. Therefore, some methods which aim to speed up the design process are needed. In this work, we propose the use of machine learning techniques to optimise NoC architecture components during the design phase. We have investigated the performance of several different ML techniques and selected the Random Forest one targeting audio/video applications. The results have shown an accuracy of up to 90% and 85% for prediction involving arbitration and routing protocols, respectively, and in terms of applications inference, audio/video achieved up to 99%. After this step, we have evaluated other classifiers for each application individually, aiming at finding the adequate one for each situation. The best class of classifiers found was the Tree-based one (Random Forest, Random Tree, and M5P) which is very encouraging, and it points to different approaches from the current state of the art for NoCs latency prediction.

Keywords Network-on-chip · Machine learning · Latency prediction · Design space exploration

✉ Jefferson Silva
jefferson.igorbr@gmail.com; jefferson.duarte@ifrn.edu.br

Extended author information available on the last page of the article

1 Introduction

The evolution of lithography process has driven an increase in circuit integration leading to more complex architectures and higher development costs, allowing that multiple circuits can be integrated into a single package, which can be also known as System on Chip (SoC) [1, 2]. This level of integration has led multiple cores in an individual silicon wafer, creating Multiple Processors Systems on Chip (MPSoCs) [3]. As the number of cores increased in a system, we have seen an increased in the possible number of communications among them. In order to improve performance, energy usage and scalability issues, Networks on Chip were employed, which use some elements from Ethernet networks, such as routers and links [4] to implement communications inside SoCs.

Since each NoC internal router component can be implemented in many different ways, finding optimised architectures relies on the understanding of a vast design space. Architectural options for NoCs stand on several possible different implementations which can range from buffering, scheduling, routing and flow control, leading to an NP-hard combinatorial problem [5]. In such a scenario, optimal architectures could only be obtained by testing each one of all possible configurations, which is computationally unfeasible. However, since applications constraints could be known before hand, optimised architectures—the ones complying with application constraints—are usually satisfactory. Even so, in order to get there, many different possible behaviours would need to be tested. This task could be accomplished using meta-heuristics, such as Genetic Algorithms [6], by traversing the design space smartly enough to satisfy the constraints while avoiding testing all architectural scenarios. Still, each architecture found by the heuristic must be simulated to verify how far it is from the desired constraints. Usually, constraints are expressed in terms of performance, energy consumption, area.

Traditional approaches rely on simulations to run the architectures and get, for instance, performance figures. However, precise results (at cycle level) could only be achieved when simulations run at lower abstraction levels, which in turn, can take too much time, due to the high computation effort needed to simulate all the components at each clock cycle. In order to overcome this situation, many approaches rely on simulations performed at higher abstraction levels. Although this brings faster simulations times, results could be compromised due to lower accuracy.

Nowadays a trend is to avoid simulations using a variety of methods, such as Analytical Modelling [7] and Machine Learning [8–10]. This tendency is satisfactory because it allows an improvement in accuracy (avoiding not obeying restrictions of time, required area or energy consumption) and a speedup in design time. Nonetheless, in all cases, designers seek to increase accuracy and to reduce the error rate. Despite this goal, NoCs have a nonlinear behaviour, which compromises the accuracy, and it is a challenge by itself. The accuracy loss could cause resource waste because the designer will use more resources than necessary aiming to obey the deadlines. Thus, it is a possible disadvantage of predictions.

Following this trend, this work proposes the usage of machine learning methods to predict latency values based on NoC architecture details. The choice by

Machine Learning methods is useful for applications where the data are difficult to model analytically and NoC has this characteristic [11]. According to Ogras, Hu and Marculescu [12] and Qian et al. [13], there are at least three main challenges in NoC design:

1. Assumptions in current queuing models (analytical models) are very tight, not supporting a wide range of traffic patterns;
2. Efficient resources management strategies;
3. Scalability challenge in NoC simulations. The authors state that simulating NoCs with more than one hundred routers demand huge computation effort, which leads to high simulation times.

Regarding the first item above, this work supports seven NoC characteristics and two application attributes to reliably predict latency, enabling the verification of more NoC configurations. The attributes are: topology, size, routing protocol, virtual channel, input buffer depth, output buffer depth, arbiter, number of packets, and required bandwidth. These last two attributes are provided by application. It impacts directly in challenge 2, as being able to explore diverse architectural configurations allows the adoption of more efficient resource management strategies. Concerning the last challenge cited above, scalability in simulations, our predictor was trained with up to 144 routers. Still, it is possible to increment this number even further, because, with our approach, it is not necessary to simulate the NoC behaviour on every single iteration of the classifier, only what is required to feed the classifier. The main goal of this work is applying the ML method, to analyse its ability to predict (correctly or not) and investigate the reason behind the accuracy predicted.

The paper is organised as follows: In Sect. 2, we present the related works. Section 3 explains the proposed framework, which is discussed in more details in Sect. 4. Section 5 presents the obtained results, followed by conclusions and future directions.

2 Related works

There are two major options to avoid having to perform all possible simulations during design space exploration:

– The use of Analytical Models; and
– The use of Machine Learning techniques (ML).

This section presents some related works and their contributions to this problem.

2.1 Analytical models

Many NoC latency models use the queuing theory, and it implies in restrictions, such as packet length and traffic follows a Poisson distribution [14].

Feng, Ge, and Wu presented an analytical model that assumes infinite buffers to predict latency in 3D NoCs [15]. It also supports multi-application mapping. It, initially, uses a Genetic Algorithm to realise the task mapping and, the resulting mapping is applied to an analytical model to verify if it obeys the designer restriction for each application. Task Graphs For Free (TGFF) [16] was used to generate synthetic graph applications, and the result of Genetic Algorithm (GA) was simulated using NoC simulator Nirgam. This model can save power consumption up by 21% and decrease the latency up to 17% when compared with random mapping.

Quian et al. developed a model that uses a G/G/1/K queuing model that supports heavy traffic, finite buffer, arbitrary packet length, and round-robin arbiter, under synthetic traffic patterns, the model achieved less than 12% of error in prediction the network saturation point. Using real applications, the proposed model can achieve 91.4–97.9% of accuracy, comparing the required time to simulate (using Booksim simulator), and the speedup was 70 $\times$ over the simulation [17].

Bhattacharya and Jha created a model that uses a G/G/1/K and M/G/1/K that can reflect the influence of buffers size, number of virtual channels, and flit size. The model was evaluated using gem5 and GARNET simulator with PARSEC benchmark suite. The obtained results show that the number of virtual channels and flits contention are inversely proportional; besides it, the authors cannot demonstrate the influence of buffers size in packet latency. Comparing the simulations results with predictions, the accuracy was superior to 85% in all scenarios [18].

Kurihara and Li proposed a model that supports four topologies, mesh, torus, hypercube, and metacube. The goal is to estimate the cost–performance ratio for these topologies. In relations to other NoC characteristics (buffers size, kind of arbiter), anyone was considered in the model. According to the obtained results, torus and metacube topologies only achieve better cost–performance when the number of cores overcomes 100 units [19].

Fischer, Fehske, and Fettweis also created a model based on queuing theory, differently to the previously cited papers, they did not take into account information such as topology or routing scheme and assumed two characteristics: infinite buffers in input channels and that the traffic follows the Poisson distribution. They consider an NoC such a hierarchical structure, in this way, is possible to split an NoC in three steps: local arrives rate, forwarding probabilities, and path delays. Thanks to this division, the model can calculate each router individually and after sum all. The accuracy, in obtained results, achieved 96% [20].

Fresse et al. dimensioned an NoC from mathematical models. To compare, they synthesised an NoC in FPGA board and compared the obtained results with the values provided by model. An explicit restriction was the use of one fixed configuration during the experiments (Mesh 2D, two virtual channels, round-robin arbiter, and XY routing algorithm). It limited the analysis of obtained results, and the authors affirmed that changing the kind of arbiter, the accuracy falls. In the worst case, this solution reached up by 88% of accuracy. Another characteristic is the low accuracy for small NoCs, and it allows us to conclude that the model cannot represent all situations adequately (do not capture the nonlinear behaviour) [21].

According to the Quian et al., analytical models cannot represent the real applications, because they assume hard restrictions, trying to relax these assumptions, they developed a new model that uses a Support Vector Machine to improve the model accuracy [22]. However, besides it, the SVM fed the analytical model. This combination achieved a result of 10% better than the previous work for the same authors—only using a mathematical model.

As mentioned earlier, the goal is to predict the value for the desired metric and an accurate inference, the number and the possibilities for each attribute are essential because it allows the model to create a realistic representation of NoC. These quoted analytical models need tight restrictions and are not capable of representing real applications accurately. In this way, the proposed framework operates with a high degree of freedom about the requirements. About the time distribution of arrived packets, our solution was trained with a variety of packet time distributions. Other difference regards on the possibility to choose among several NoC topologies sizes. The proposed solution was trained with two topologies ranging from 4 to 144 routers. Each author adopted a set of information to feed his model and to predict the desired metric, but anyone of the quoted paper informed the use of seven and two NoC characteristics and applications, respectively. Thus, our approach supports a higher number of applications because the proposed model supports more options for each attribute than the presented in the quoted papers, such as routing protocol which this paper handles six different protocols.

2.2 Machine learning-based solutions

Machine Learning (ML) techniques are commonly used for design space exploration purposes. For instance, Chen et al. use an ML technique to speed up networks up to 8 routers; they used RankBoost method to find the best configuration and creates a configuration rank to choose the best one, based on a defined metric threshold. However, this work did not focus on NoC DSE, but in CPU DSE [23].

Few works use ML methods to focus on regular NoC topologies, while the opposed happen for irregular ones. Specifically, about irregular topologies, Chou and Marculescu created a platform focused in user experience to explore the design space using Machine Learning techniques to cluster the traces from similar users and, for each cluster, an algorithm for automatically architecture generation. This algorithm in a second phase implements an analytical model and takes into consideration many characteristics such as architecture template (resources of computation, communication, protocols), applications specification (task graph, deadlines, power consumption), and user traces (restrictions about latency, power consumption) [24].

In this direction, Zhang et al. used a congestion matrix to predict the worst packet latency in NoCs. According to the authors, the worst-case traffic model can adequately reflect the dynamic behaviours of packet switching networks. To find the best configuration (the solution supports seven attributes) for the performance, they used a local search algorithm to explore NoC configurations; the platform also provides an area and power consumption estimations, generated using ORION [25] and DSENT [26] tools. The results showed that the bigger is the NoC size, lesser is the solution accuracy. They also presented accuracy results over injection rates. In this case, for lower injection rates, such as from 0.1 to 0.3, the accuracy found was less than 70% [27].

Aiming at obtaining an adequate NoC configuration, Ayari et al. developed a hypervolume-based approach to refining the DSE. They used as metric the load variation, power consumption, and communication costs. Because it is a multi-objective solution, the authors used a Reduced Pareto front (RPF) to facilitate the designer solution choice. The proposed approach had two stages: first, Pareto optimal solutions are clustered using K-means algorithm, and, as the second stage, a subset of solutions that maximises the hypervolume is selected through a genetic algorithm [28].

Different from the other approaches, our solution supports a significant number of routing and arbiters implementations. This is important since Fresse et al. argue that these characteristics may lead to improvements on the prediction errors due to the nonlinearity behaviour of communications [21]. Compared with irregular topologies, our work presents some advantages, such as minor design cost and, hence, higher possibility of reuse. Minor design cost happens because the architectures found can be shared among several applications. Contrasting with other approaches, our work is more flexible, not being tied to specific network sizes or target applications.

3 Proposed approach

This work aims to expand the use of classifier techniques to NoCs architectural characteristics in order to maximise its overall accuracy. The goal of this classifier is to predict average latency values based on NoCs and traffic pattern characteristics. For reaching this goal, we defined a sequence of work as presented in Fig. 1.

Figure 1 defines the workflow. The first step is to perform simulations and to gather the results. In our case, we stopped this phase when we aggregated 496 instances. The data follow the requirements of Weka tool dataset. In the sequence, we identified all classifiers which operate in a regressive way implemented in the tool, and we evaluated all of them. We present more details for each step in the next subimages.

3.1 Machine learning classifiers

Seeking out to predict accurately, the solution uses a regressive supervised method. We evaluated all regressive classifiers already implemented in Weka software. The list is presented below:

- *Isotonic regression* Selects the attribute that results in the lowest squared error;
- *Linear regression* It works by estimating coefficients for a line or hyperplane that best fits the training data;

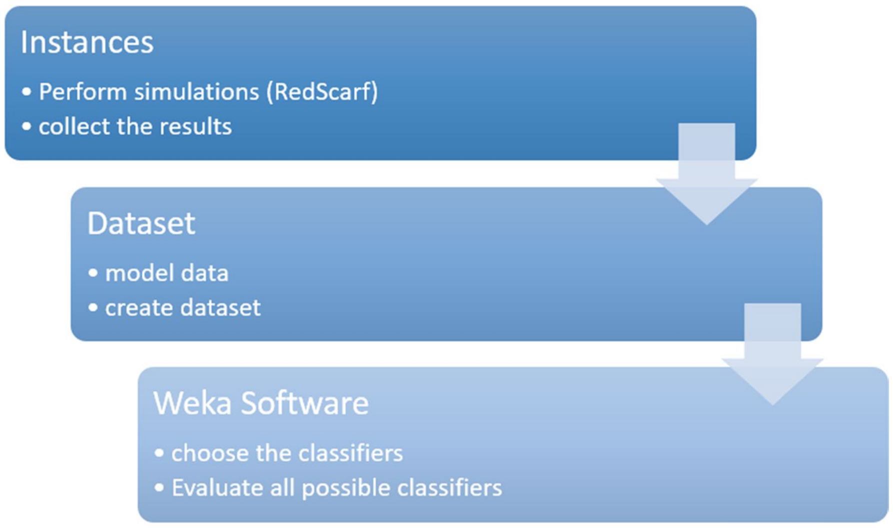


Fig. 1 Sequence of this approach

- *MLP regressor* Trains a Multilayer Perceptron (MLP) with one hidden layer by minimising the given loss function plus a quadratic penalty with the Broyden-Fletcher-Goldfarb-Shanno method;
- *MLP* An MLP consists of at least three layers of nodes. Except for the input nodes, all nodes using a nonlinear activation function. MLP uses a technique called backpropagation for training;
- *RBF network and RBF regressor* An RBFN performs classification by measuring the inputs similarity to examples from the training set. When we want to classify a new input, each neuron computes the Euclidean distance between the input and its prototype;
- *SMOreg* Implements the Support Vector Machine (SVM) for regression;
- *IBk* *K*-nearest neighbours classifier. It uses the Euclidean distance to calculate the distance between the instance and k neighbours;
- *M5Rules* Generates a series of M5 trees, where only the "best" (highest coverage) leaf/rule is retained from each tree. At each stage, the instances covered by the best rule are removed from the training data before generating the next tree;
- *M5P* This technique works as ordinary decision trees with linear regression models at the leaves that predict the value of observations that reach the leaf. The nodes of the tree represent variables and branches represent split values;
- *Random forest* The Random Forest algorithm consists of a random collection of decision trees. Hence, this algorithm is just an extension of the decision tree algorithm.

Table 1 All possibilities to NoC design space exploration in this work

Characteristic	Number of possibilities
Topology	2
Size	15
Routing protocol	6
Virtual channels	33
Input buffer depth	28
Output buffer depth	33
Arbiter type	4
Total	21,954,240

In order to use a well-known platform for others to be able to replicate our experiments, we have used the Weka[1] software, version 3.8. Each dataset represents an application. The statistical test Student t test was applied to validate the results [29].

All needed simulations were performed on RedScarf Simulator [30]. It operates in Register Transfer Level (RTL) and has high accuracy and also has support to manipulate seven NoC attributes. Models for latency and flow rate follow the model‘s Dally and Towles [31].

3.2 Datasets

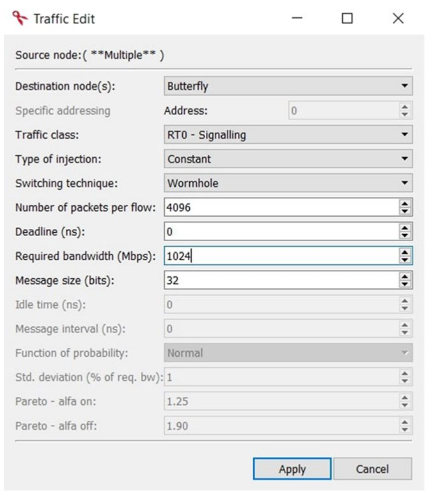
Audio/video dataset contains 496 instances, each instance mapped to a specific NoC configuration plus two specific information about the application, precisely number of packets and required bandwidth, and the latency's value. Traffic generation was based in [32], and the communication distribution uses four models: Bit-reversal, Perfect Shuffle, Butterfly, and Transpose Matrix.

An instance has a network size between 2×2 to 12×12. These sizes were needed because the nonlinear variation in some circumstances, such as latency using non-deterministic routing algorithm, due to the occurred contention and possibility of deadlocks. Each instance represents a performed simulation on RedScarf simulator.

In addition to NoC attributes, two pieces of information about the application traffic pattern were used, the number of packets (NoP) and required bandwidth (RB). For NoP, values were 128, 1024, and 8192 per communication flow and for RB, 64, 512, and 1024 Mbps. All experiments were performed using all values for NoP and RB. This range is necessary to improve the accuracy of the classifiers.

There are three other datasets, one for each tested application: signalling, read/write, and block transfer. They contain 1,488 instances each. This number was reached varying NoC and application parameters. Similar to audio/video dataset, the same NoC values to attributes were used. Besides, applications characteristics were expanded. Number of packets was evaluated with 128, 1024, 2048, and 8192;

[1] https://www.cs.waikato.ac.nz/ml/weka/.



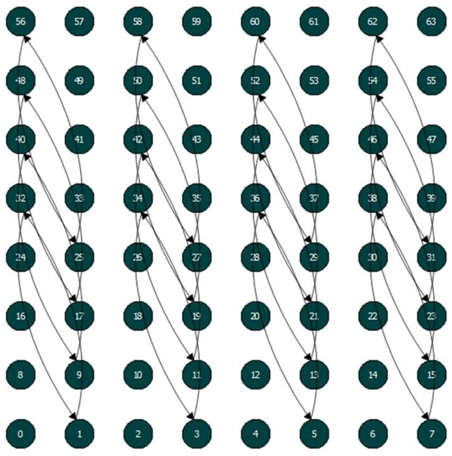
Traffic Edit

Source node:(**Multiple**)

Destination node(s):	Butterfly
Specific addressing	Address: 0
Traffic class:	RT0 - Signalling
Type of injection:	Constant
Switching technique:	Wormhole
Number of packets per flow:	4096
Deadline (ns):	0
Required bandwidth (Mbps):	1024
Message size (bits):	32
Idle time (ns):	0
Message interval (ns):	0
Function of probability:	Normal
Std. deviation (% of req. bw):	1
Pareto - alfa on:	1.25
Pareto - alfa off:	1.90

Apply Cancel

(a) Traffic configuration in RedScarf.



(b) Mesh 8x8 topology with the specification of traffic among routers.

Fig. 2 Example of NoC configuration in RedScarf

required bandwidth was used four values: 64, 512, 1024, and 4096. The intention of using wide datasets was to achieve better training for classifiers.

All configurations for all characteristics gave us the total number of possibilities for a NoC, as shown in Table 1.

By analysing the data on Table 1, one can conclude that there are a massive number of possibilities for each NoC and, hence, a high number of simulations to deplete all options. Assuming each simulation executing in ten seconds, 60,984 h will be necessary to simulate everything (sequential simulations). Therefore, the DSE is huge and is almost unfeasible to evaluate all possibilities in an affordable time. Besides, there is another problem: each change made may impact in other attributes, leading designers to be forced to perform still more simulations and analysis to know the NoC final performance, which requires even more time. Based on this scenario, the values in Table 1 justify the use of AI methods for predictions.

Dataset was formatted according to Weka software.[2] Each experiment execution was repeated ten times, this value is a suggestion of Weka developers. Information about the RedScarf simulator is presented in the next section.

3.3 RedScarf simulator

RedScarf simulator implemented using System-C language with supports RTL simulation and multi-threading operation [33]. It supports real-time and non real-time applications. Figure 2 exhibits RedScarf configuration.

[2] https://www.cs.waikato.ac.nz/ml/weka/downloading.html.

Table 2 *t*-Test statistical test execution over all classifiers

Dataset	(1)		(2)		(3)		(4)		(5)	
Latency prediction	0.95 ± 0.06		0.62± 0.10	•	0.71 ± 0.08	•	0.89 ± 0.07	•	0.89 ± 0.07	•
	(6)		(7)		(8)		(9)		(10)	
	0.51 ± 0.24	•	0.90 ± 0.06		0.89 ± 0.06	•	0.85 ± 0.09	•	0.93 ± 0.05	

Three classifiers obtained the same statistical significance

∘, • Statistically significant improvement or degradation

Figure 2 presents a topology with size equal to 8 × 8 and its traffic following Butterfly distribution. Figure 2a specifies the created traffic, where the designer can configure the traffic characteristic, some attributes are: distribution pattern of traffic, application (traffic class), number of packets, deadline (for real-time applications), required bandwidth (also is known as packet injection rate), and message size. Figure 2b includes routers and its traffic (each arrow represents a communication flow).

The dataset was modelled aiming to represent the implemented attributes in RedScarf. The focus of simulator is performance simulation. Thus, it does not provide any information about the demanded area or power consumption. This tool offers a high precision because it operates in Register Transfer Level; despite this, it takes a long time to perform each simulation, which may make a high number of simulations unfeasible due to the high demanded time.

Next section will provide information about the Experiments.

4 Experiments and results

For this research, the results were generated in two steps. First, it was necessary to simulate several NoC configurations, focusing on the average latency of packets. Each experiment was executed ten times. This value is the default in the simulator and is used by simulator's authors in their tests. These simulations were made using the RedScarf simulator running on a Debian 9 Linux.

After capturing the simulation results, the data were formatted in the Weka dataset file. Each attribute is related to a specific NoC characteristic. Thus, testing different NoCs accounts for changing any of the attributes (or even more than one) at a time. The next step was to evaluate all quoted classifiers listed in Sect. 3. Table 2 shows the student test result, and this statistical test analysed the accuracy information, not latency's result.

In Table 2, one can observe that three classifiers have obtained equivalent results. Table 3 presents the used configuration for each classifier; these values are the default in Weka software. Exploiting each hyperparameter of each technique is out of scope for this paper.

As can be seen in Table 2, three classifiers achieved the same performance. Thus, aiming to choose the best classifier, we analysed the mean absolute error (MEA) as the second metric. Table 4 presents the results for this metric for the best classifiers.

Table 3 Classifiers configuration used during the experiments

(1)	functions.IsotonicRegression " 1679336022835454137
(2)	functions.LinearRegression '-S 0 -R 1.0E-8 -num-decimal-places4' -3364580862046573747
(3)	functions.MLPRegressor '-N 2 -R 0.01 -O 1.0E-6 -P 1 -E 1 -S 1 -L functions.loss.SquaredError -A functions.activation.ApproximateSigmoid' -4477474276438394655
(4)	functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779
(5)	functions.RBFNetwork '-B 2 -S 1 -R 1.0E-8 -M -1 -W 0.1' -3669814959712675720
(6)	functions.RBFRegressor '-N 2 -R 0.01 -L 1.0E-6 -C 2 -P 1 -E 1-S 1' -7847474276438394611
(7)	functions.SMOreg '-C 1.0 -N 0 -I \"functions.supportVector.RegSMOImproved -T 0.001 -V -P 1.0E-12 -L 0.001 -W 1\" -K \"functions.supportVector.Puk -O 1.0 -S 1.0 -C 250007\"" -7149606251113102827
(8)	lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance-R first-last\\\"\""-3080186098777067172
(9)	trees.M5P '-M 4.0' -6118439039768244417
(10)	trees.RandomTree '-L 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698

Table 4 Obtained values of mean absolute error metric for each of three classifiers

Classifier	Mean absolute error (%)
Isotonic regression	21
SMOreg	13
Random Tree	9

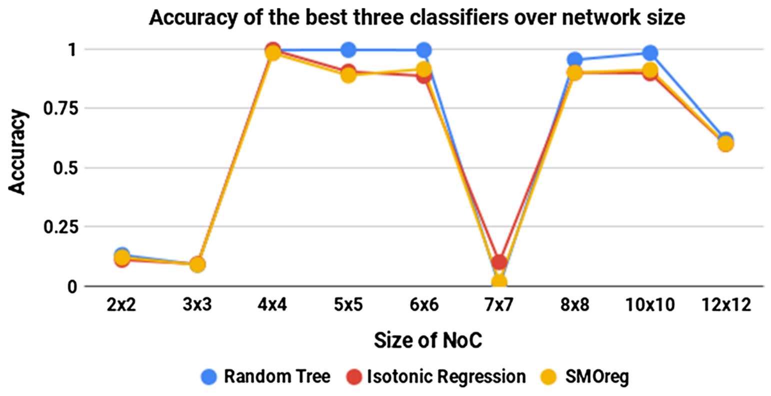


Fig. 3 Comparison between achieved accuracy using Isotonic Regression, Random Tree, and SMOreg for a range of NoC sizes

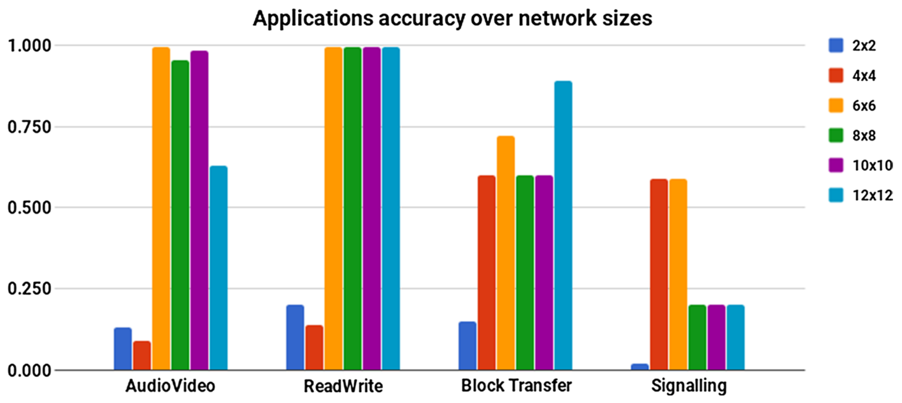


Fig. 4 Achieved accuracy for a range of NoC sizes using four applications

Table 4 shows percentage of error for each of three classifiers. Random Tree method reached the minor error, without penalising accuracy.

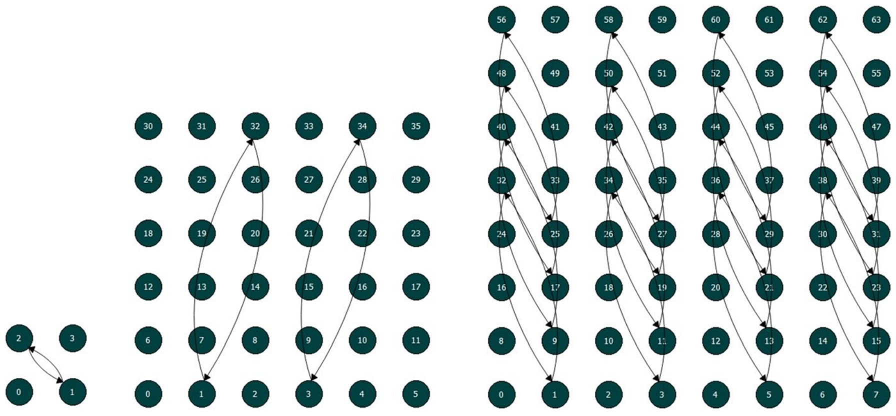
Figure 3 presents an analysis of accuracy for each NoC size.

Figure 3 shows the accuracy of the Random Tree, Isotonic Regression, and SMOreg (Support Vector Machine for regressive problems) for several network sizes. Besides the high variation, from zero to near 1, the error rate sustained lower values. Analysing the accuracy for size seven, the classifiers almost missed all instances, but with a smaller error (less than 10%). In five sizes, out of 9 in total, the classifier overcame 95% of accuracy. This kind of behaviour is expected because of the intrinsic behaviour of NoCs, such as contentions. Comparing them—all of them using the hyperparameter presented in Table 3—is possible to notice that the techniques had difficulty to infer the latency in some sizes. This allows us to conclude that the problem is not the adopted techniques by themselves rather the intrinsic behaviour of NoCs, which is nonlinear and may get contentions depending on application or traffic pattern distribution. For now on, all experiments were conducted using Random Tree classifier. Figure 4 shows the classifier accuracy for four applications.

As can be seen in Fig. 4, read/write application achieved an accuracy higher than 99% from 6×6 to 12×12 networks. On the other hand, signalling application had the worst accuracy; only two NoC sizes achieved over 50% of accuracy. Audio/video application had similar results as shown in Fig. 3.

For all applications, in 2×2 network size, the classifier reached up to 20% of accuracy. Although the training process has the same number of instances for each network sizes, this result indicates that the Decision Tree used could not predict the latency values correctly for small NoCs. This could happen due to the nonlinear behaviour of networks, imposing constraints for the learning process. Comparing the application distribution as presented in Fig. 5, we can see some differences.

Figure 5 shows three traffic patterns, one for each NoC size, 2×2, 6×6, and 8×8. In all cases, the application was mapped using butterfly distribution. Thus, the number of communications changes from two to thirty-two in Fig. 5c. This change



(a) Butterfly distribution for 2x2 mesh. **(b)** Butterfly distribution for 6x6 mesh. **(c)** Butterfly distribution for 8x8 mesh.

Fig. 5 Example of NoC applications using butterfly distribution, but with three different sizes

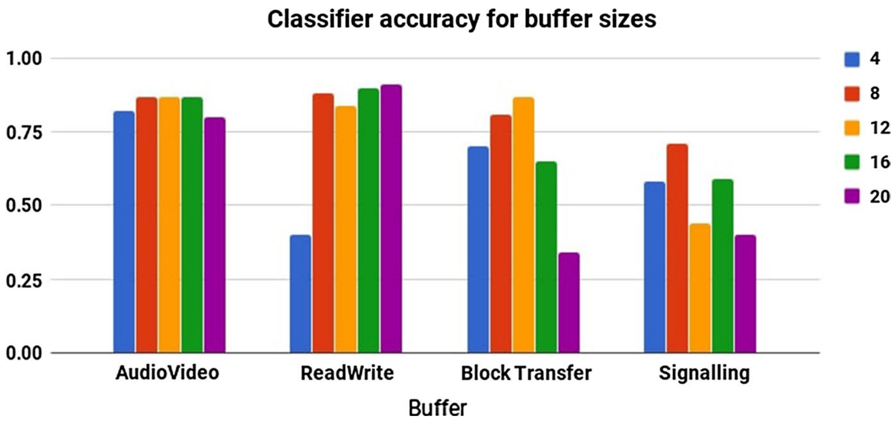


Fig. 6 Classifier accuracy for buffer sizes using four applications

could impact directly in performance because it increases the possibility to occur contention, saturate the channels, for instance. Therefore, this NoC characteristic becomes more complex the task of the ML method to understand and generate a suitable model to a wide range of NoC sizes. Figure 6 presents the classifier accuracy for buffer sizes.

Figure 6 shows that the used classifier achieved, in two of five cases for signalling application, accuracy up to 45%. For block transfer application, buffer size equal to 16 and 20 had the worst accuracy, less than 70%, and 40%. In other cases, the accuracy overcame 70%. Read/write application achieved an accuracy greater than 80% in four scenarios. These results corroborate with the expected performance when the designer increases the buffer sizes: at a point, the performance

Table 5 Best classifier for each application

Application	Method	CC	MAE	RAE (%)
Signalling	Random forest	0.9214	970.2602	14.3569
Read/write	Random Tree	0.8753	1012.8454	14.0498
Block transfer	M5P	0.906	466.6563	11.7552

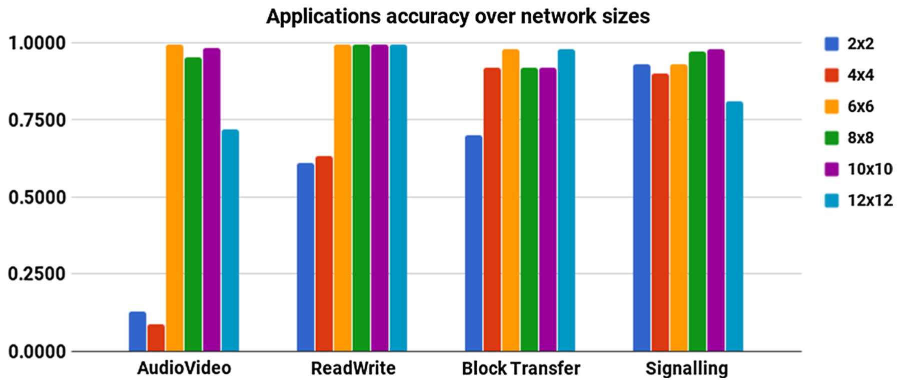


Fig. 7 Best classifiers accuracy to some NoC sizes using four applications

increments proportionally, but, after reaching the limit, increases the buffer take to waste of resources. This behaviour is unique for each application, each one has its resource usage. Therefore, the variation presented in Fig. 6 is habitual. Taking Fig. 2b as example and assuming the use of XY routing protocol [34], the router 25 forwards four communications: from router 32 to 1; from 40 to 9; from 48 to 17; from 56 to 25. Thus, this router needs the buffer to store the flits while it cannot forward ahead temporarily. However, on the other hand, router 0 does not need any buffer, because, in butterfly distribution, it does not receive any communication.

Based on previous results, it was analysed the possibility of usage of other classifier and Table 5 shows the best classifier for each application analysed.

Although each application requires a specific technique, all belongs to Decision Trees class. So, for the suggested approach, this class of methods is adequate. Figure 7 shows the accuracy using the best classifier for each application.

Analysing each application separately, it is possible to notice that all three applications had an increase in accuracy. Signalling application overcame 70% of accuracy on overall. Worst scenario for read/write application was 2×2, but with the most adequate classifier, it overcame 55% of accuracy. Block transfer application in five, out of six in total, overcame 92% of accuracy. Despite this improvement, audio/video application still obtained a lower result for sizes equal to 2×2 and 4×4. Thus, even the best classifier for this application could not achieve high accuracy for both scenarios. Both sizes gathered the lowest accuracy for also Read/Write application, which demonstrates the lower requirement for communication and the disability of classifier to handle both sizes for these applications.

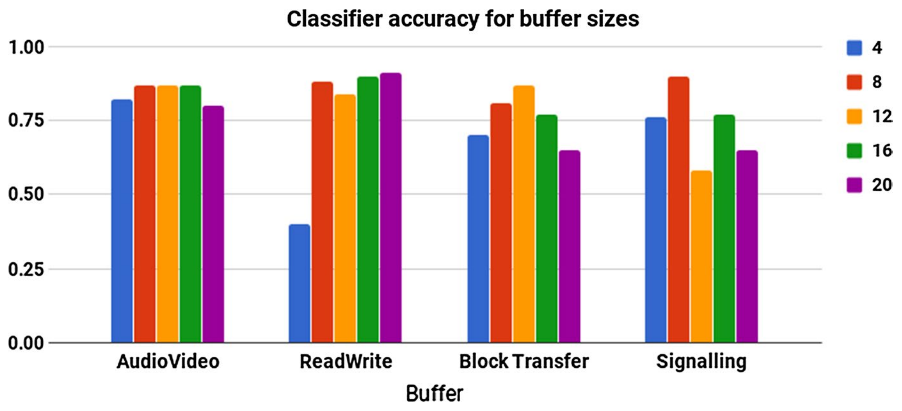


Fig. 8 Best classifiers accuracy for buffer sizes using four applications

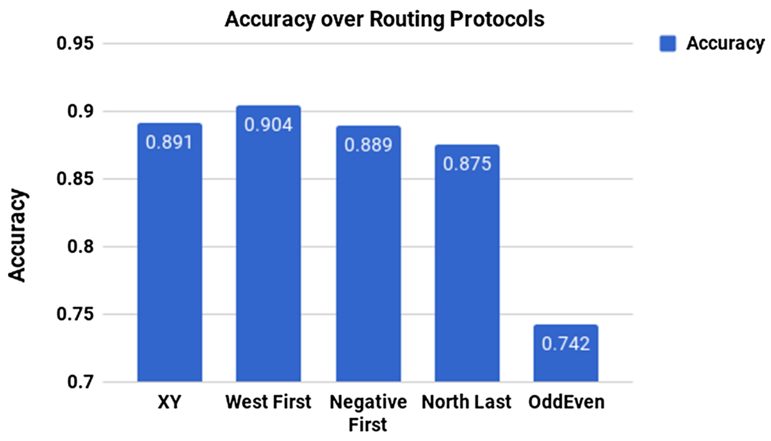


Fig. 9 Accuracy of classifier for each routing protocol supported

However, for all applications, the accuracy increased for sizes equal to or higher than 6 × 6; it represents that the classifier could model this behaviour correctly.

Below, there is Fig. 8. It presents the accuracy for buffer sizes using the adequate classifier.

As can be seen, there was an increase in accuracy for three applications (all except audio/video) when adopted the best classifier for each application. For the signalling application, the results overcame 80% of accuracy at the worst scenario and for other scenario overcame 93% of accuracy. Block transfer application had a similar growth, having like the worst case when buffer size is equal to 4 (less than 76% of accuracy). Read/write application had an accuracy higher than 50% in the worst case (also when buffer size is 4) and overcame 91% in four of five cases. Although the error rate appears to be significant, the error was

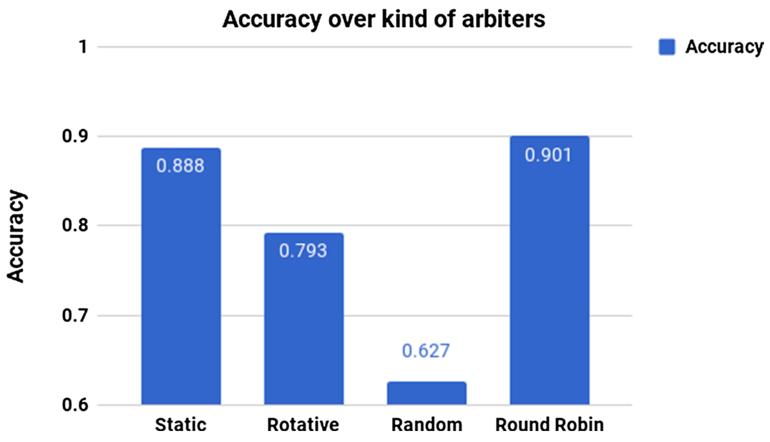


Fig. 10 Accuracy of classifier for each arbiter supported

always between 11 and 14%. In this way, it is possible to tune the values based on this error rate. Figure 9 presents the accuracy for all routing protocols supported.

By observing Fig. 9, it is possible to notice that accuracy for OddEven protocol is less than 80%. This can be explained by the adaptive nature of the protocol that could lead to deadlocks, therefore making the prediction harder than for the other protocols. For the other four protocols, accuracy overcame 85%, which can be justified because the deterministic behaviour of these routing protocols. Other similar situation is shown in Fig. 10.

Figure 10 walks in the same way that the previous image: classifier had difficulty in predicting adaptive or random situations. More specifically, random arbiter was the worst scenario, reaching less than 70% of accuracy. However, for the other types of arbiters, the accuracy overcame 79%.

An option to improve the accuracy for all cases is the use of a committee of classifiers. In this direction, committee classification units could combine the outputs of two or more classifiers to generate a unique result [35].

5 Conclusions and future works

This work presented research about latency prediction for Networks on Chip communication architectures, based on the characteristics of the architectures and applications. Artificial intelligence techniques were employed to build solutions able to predict latencies figures up to 99% in accuracy.

Networks on Chip were evaluated with mesh topologies ranging between 4 and 144 nodes. The experiments were conducted in two steps, taking simulation results to train the predictor.

Random Tree was used as Latency Predictor for nine attributes. The instances were collected from the RedScarf simulation tool. In total, four real applications were analysed.

Since good accuracy could be achieved by the predictor, it can be tuned to help speeding up the design space exploration tools for NoC-based communication architectures. This is an essential contribution of the paper due to the novel approach proposed to predict the average latency for distributed applications.

As future works, we intend to extrapolate the experiments for other performance metrics and to analyse the usage of a committee of classifiers to improve even more the accuracy. Another possible direction relies on the use of heuristics to select the adequate characteristics for target architectures, aiming at reducing the needed resources and time to get optimised solutions.

Acknowledgements This research was supported by High Performance Computing Center at UFRN (NPAD/UFRN).

References

1. Zeferino CA (2003) Redes-em-chip: arquiteturas e modelos para avaliação de área e desempenho
2. Jain K, Singh SK, Majumder A, Mondai AJ (2015) Problems encountered in various arbitration techniques used in NoC router: a survey. In: 2015 International Conference on Electronic Design, Computer Networks and Automated Verification (EDCAV), IEEE, pp 62–67
3. Wolf W, Jerraya AA, Martin G (2008) Multiprocessor system-on-chip (mpsoc) technology. IEEE Trans Comput Aided Des Integr Circuits Syst 27(10):1701–1713
4. Bjerregaard T, Mahadevan S (2006) A survey of research and practices of network-on-chip. ACM Comput Surv (CSUR) 38(1):1
5. Kahruman-Anderoglu S, Buchanan A, Butenko S, Prokopyev OA (2016) On provably best construction heuristics for hard combinatorial optimization problems. Networks 67(3):238–245
6. Elbeltagi E, Hegazy T, Grierson D (2005) Comparison among five evolutionary-based optimization algorithms. Adv Eng Inf 19(1):43–53
7. Ost L, Mandelli M, Almeida GM, Moller L, Indrusiak LS, Sassatelli G, Benoit P, Glesner M, Robert M, Moraes F (2013) Power-aware dynamic mapping heuristics for NoC-based MPSoCs using a unified model-based approach. ACM Trans Embed Comput Syst (TECS) 12(3):75
8. Tei YZ, Marsono MN, Shaikh-Husin N, Hau YW (2013) Network partitioning and ga heuristic crossover for NoC application mapping. In: 2013 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, pp 1228–1231
9. Faragardi HR, Shojaee R, Yazdani N (2012) Reliability-aware task allocation in distributed computing systems using hybrid simulated annealing and tabu search. In: 2012 IEEE 14th International Conference on High Performance Computing and Communication and 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), IEEE, pp 1088–1095
10. Wu C, Deng C, Liu L, Han J, Chen J, Yin S, Wei S (2017) A multi-objective model oriented mapping approach for NoC-based computing systems. IEEE Trans Parallel Distrib Syst 28(3):662–676
11. Sze V, Chen Y-H, Emer J, Suleiman A, Zhang Z (2017) Hardware for machine learning: challenges and opportunities. In: IEEE Custom Integrated Circuits Conference (CICC), IEEE 2017, pp 1–8
12. Marculescu R, Hu J, Ogras UY (2005) Key research problems in NoC design: a holistic perspective. In: Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2005. CODES+ ISSS'05. IEEE, pp 69–74
13. Qian Z, Bogdan P, Tsui C-Y, Marculescu R (2016) Performance evaluation of NoC-based multicore systems: from traffic analysis to NoC latency modeling. ACM Trans Des Autom Electron Syst (TODAES) 21(3):52
14. Ogras UY, Bogdan P, Marculescu R (2010) An analytical approach for network-on-chip performance analysis. IEEE Trans Comput Aided Des Integr Circuits Syst 29(12):2001–2013
15. Feng G, Ge F, Wu N (2015) Balancing 3D network-on-chip latency in multi-application mapping based on m/g/1 delay model. In: Proceedings of the World Congress on Engineering and Computer Science, vol 1

16. Dick RP, Rhodes DL, Wolf W (1998) Tgff: task graphs for free. In: Proceedings of the Sixth International Workshop on Hardware/Software Codesign. (CODES/CASHE'98), IEEE, pp 97–101
17. Qian Z, Juan D-C, Bogdan P, Tsui C-Y, Marculescu D, Marculescu R (2014) A comprehensive and accurate latency model for network-on-chip performance analysis. In: 2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE, 2014, pp 323–328
18. Bhattacharya D, Jha NK (2017) Analytical modeling of the smart NoC. IEEE Trans Multi Scale Comput Syst 3(4):242–254
19. Kurihara T, Li Y (2016) A cost and performance analytical model for large-scale on-chip interconnection networks. In: 2016 Fourth International Symposium on Computing and Networking (CANDAR), IEEE, pp 447–450
20. Fischer E, Fehske A, Fettweis G P et al (2012) A flexible analytic model for the design space exploration of many-core network-on-chips based on queueing theory. In: Proceedings of the Fourth International Conference on Advances in System Simulation, ser. SIMUL, Citeseer
21. Fresse V, Combes C, Payet M, Rousseau F (2016) NoC dimensioning from mathematical models. Int J Comput Digit Syst (IJCDS) 5(2):135–145
22. Qian Z, Juan D-C, Bogdan P, Tsui C-Y, Marculescu D, Marculescu R (2013) Svr-NoC: A performance analysis tool for network-on-chips using learning-based support vector regression model. In: Proceedings of the Conference on Design, Automation and Test in Europe, EDA Consortium, pp 354–357
23. Chen T, Guo Q, Tang K, Temam O, Xu Z, Zhou Z-H, Chen Y (2014) Archranker: A ranking approach to design space exploration. In: 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), IEEE, pp 85–96
24. Chou C-L, Marculescu R (2009) User-centric design space exploration for heterogeneous network-on-chip platforms. In: Proceedings of the Conference on Design, Automation and Test in Europe, European Design and Automation Association, pp 15–20
25. Kahng A B, Li B, Peh L-S, Samadi K (2009) Orion 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In: Design, Automation and Test in Europe Conference and Exhibition, 2009. DATE'09, IEEE, pp 423–428
26. Sun C, Chen C-HO, Kurian G, Wei L, Miller J, Agarwal A, Peh L-S, Stojanovic V (2012) Dsent-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In: 2012 Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS), IEEE, pp 201–210
27. Zhang M, Shi Y, Zhang F, Liu Z (2016) Comrance: a rapid method for network-on-chip design space exploration. In: Green and Sustainable Computing Conference (IGSC0) 2016 Seventh International, IEEE, pp 1–8
28. Ayari R, Nikdast M, Hafnaoui I, Beltrame G, Nicolescu G (2017) Hypap: a hypervolume-based approach for refining the design of embedded systems. IEEE Embed Syst Lett 9(3):57–60
29. Hsu P (1938) Contribution to the theory of "student's" *t*-test as applied to the problem of two samples, Statistical Research Memoirs
30. SILVA EA Redscarf: ambiente para avaliação de desempenho de rede-em-chip, Trabalho Técnico Científico de Conclusão de Curso, Universidade do Vale do Itajaí
31. Dally WJ, Towles BP (2004) Principles and practices of interconnection networks. Elsevier, Amsterdam
32. Tedesco L, Mello A, Garibotti D, Calazans N, Moraes F (2005) Traffic generation and performance evaluation for mesh-based NoCs. In: Proceedings of the 18th Annual Symposium on Integrated Circuits and System Design, ACM, pp 184–189
33. da Silva EA, Menegasso D, Vargas S, Zeferino CA (2017) Redscarf: a user-friendly multi-platform network-on-chip simulator. In: VII Brazilian Symposium on Computing Systems Engineering (SBESC). IEEE 2017, pp 71–78
34. Rao MV, Krishna TR, Siddhartha S, Prathyusha KS (2017) A load balance aware xy routing methodology for NoC architectures. Adv Appl Math Sci 17(1):251–270
35. Aksela M (2003) Comparison of classifier selection methods for improving committee performance. Multiple Classifier Systems, pp 159–159

Affiliations

Jefferson Silva[1] · Márcio Kreutz[2] · Monica Pereira[2] · Marjory Da Costa-Abreu[2]

Márcio Kreutz
kreutz@dimap.ufrn.br

Monica Pereira
monicapereira@dimap.ufrn.br

Marjory Da Costa-Abreu
marjory@dimap.ufrn.br

1 Federal Institute of Education, Science and Technology of Rio Grande do Norte (IFRN), Natal, RN, Brazil

2 UFRN, Natal, Brazil